Benchmarking Software dan *Website Report* untuk Memudahkan Kategorisasi *Device* Berbasis *Android* Berdasarkan Performa

Michael Christian Tjandra[1], Justinus Andjarwirawan[2], Henry Novianus Palit[3]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121-131, Surabaya 60236
Telp (031) – 2983455, Fax. (031) - 8417658
E-Mail: glicohonda@gmail.com, justin@petra.ac.id[2], hnpalit@petra.ac.id[3]

ABSTRAK

Pengimplementasian *Benchmarking Software* dan *Website Report* untuk Memudahkan Kategorisasi *Device Android* Berdasarkan Performa

Maraknya *device* dengan *platform Android* membuat masyarakat dapat memiliki banyak alternatif pilihan *device* dari berbagai *vendor*. Banyak faktor yang perlu dipertimbangkan seorang *user* untuk akhirnya dapat memilih *mobile device* di pasar yang dapat memenuhi kebutuhannya. Diantaranya adalah dengan memperhatikan segi *performance*, *size*, dan *price*.

Aplikasi *benchmark* akan menguji beberapa sektor utama dari *device* yaitu *CPU computation, RAM*, *JavaScript performance*, *Graphics*, dan *Storage* dengan menjalankan metode pengujian sesuai bagiannya. Selain aplikasi, disediakan sebuah *website report* agar layanan dan data hasil pengujian yang telah diunggah dapat mudah diakses oleh khalayak umum dengan tujuan mempermudah klasifikasi *mobile device*.

Berdasar hasil dari pengujian, aplikasi dapat berjalan sepenuhnya dan dapat menjalankan semua fungsi - fungsinya. Proses pemberian *score* sudah relevan dan mampu memberikan gambaran secara umum perihal kemampuan dari *mobile device* yang beredar dipasaran. Pada beberapa *device* dengan versi *Android* terbaru, *permission* untuk penggunaan *external storage* dari aplikasi menjadi kendala untuk melakukan pengujian *storage*.

Kata Kunci: *Android, Benchmark, Website Report.*

ABSTRACT

Benchmarking Software and Website Report Implementation for Easier Android Devices Categorization Based on Performance

The popularity of Android based device could lead into an abundant amount of device choices from many vendors. There's a lot factors that one needed to consider before they can decide one device on the market that fully coping their needs. Some of whice are device's performance, size, and price.The benchmark application will put the device into trial which consists of CPU computation, JavaScript performance, Graphics, and Storage capability. Application aside, there will be a web service intended to provide service for masses in an easier and a simpler way.

Based on the test results, the application can run well and can perform all of its functions. Score output are relevant and can give the user general view about the performance from the device on the market. On several later iteration of android version, usage permission for external storage are very limited. Hence, DriveSpeed2 can't be run because the application couldn't write on external storage freely.

***Keywords*:** *Android, Benchmark, Website Report.*

1. PENDAHULUAN

Benchmarking adalah proses evaluasi *performance* dari suatu sistem pada kondisi tertentu. *Benchmarking* dilakukan dengan menjalankan sebuah atau kumpulan program yang dinamakan *benchmarking tools*. Program tersebut akan menjalankan operasi-operasi tertentu untuk menguji kemampuan dari suatu sistem. *Benchmarking tools* dalam dunia komputasi secara umum menguji beberapa aspek dalam komputer seperti kemampuan CPU (*HyperPi*, *Prime95*, *CPU-M*), *storage* (CrystalDiskMark), *graphics* (*3DMark*, *Unigine Heaven*, *Maxon Cinebench*), maupun sistem secara keseluruhan (*PCMark*, *SiSoftware Sandra*, *NovaBench*, *PassMark Performance Test*) [8].

Dalam konteks *mobile device benchmarking*, proses *benchmarking* dilakukan dengan menjalankan aplikasi pada *gadget* yang akan diuji. Aplikasi tersebut akan menjalankan beberapa metode *benchmarking* yang akan menguji kinerja dari beberapa aspek pada *device* seperti kemampuan *CPU arithmetic*, *graphic* (*2D* dan *3D*), *browser engine*, dan *storage input* dan *output*. Aplikasi *benchmarking* terkemuka saat ini seperti *AnTuTu*, *PassMark*, dan *AndroBench* memang sudah mampu memberikan angka-angka yang menunjukkan kemampuan dari berbagai *device* yang ada saat ini. Namun hasil tersebut tidak dapat membantu masyarakat Indonesia dalam memilih *device* yang tepat dikarenakan kebanyakan tidak terdapat *device* dari *vendor* lokal dan tidak disertai dengan harga dan *release date*.

2. TINJAUAN PUSTAKA

2.1 LINPACK

LINPACK merupakan sebuah *software library* untuk melakukan komputasi aljabar linier pada komputer *digital*. Pertama kali ditulis dalam bahasa *FORTRAN* oleh Jack Donggara, Jim Bunch, Cleve Moler, dan Gilbert Stewart dan ditujukan untuk *supercomputer* sekitar tahun 1970 dan awal 1980.

LINPACK menggunakan *library* BLAS (*Basic Linear Algebra Subprograms*) yang merupakan dasar untuk melakukan perhitungan vektor dan matriks dasar. Level 1 BLAS melakukan operasi vektor-vektor. Level 2 BLAS melakukan operasi matriks-vektor. Level 3 BLAS melakukan operasi matriks-matriks.

LINPACK digunakan untuk *benchmark* dan menentukan peringkat *supercomputer* pada daftar TOP500.

Seiring berjalannya waktu, ruang lingkup pengujian *LINPACK* semakin luas. Laporan pengujian menjelaskan kemampuan menyelesaikan permasalahan *matrix* Ax- b pada *64-bit floating-point arithmetic* pada tiga tingkat permasalahan yakni 100 kali 100 (*inner loop optimization*), 1000 kali 1000 (*three loop optimization*) dan *scalable parallel* [2].

2.2 SciMark2

Scimark2 merupakan salah satu benchmarking yang melakukan perhitungan kemampuan mengolah angka untuk aplikasi *scientific* dan *engineering*. Scimark2 terdiri dari lima *computational kernel* yakni *FFT, Gauss-Seidel relaxation, Sparse matrix-multiply, Monte Carlo integration,* dan *dense LU factorization*.

Kernel tersebut dipilih untuk menyediakan indikasi seberapa baik JVM (*Java Virtual Machine*) menjalankan aplikasi dengan menggunakan algoritma tersebut. Ukuran dari soal telah ditentukan untuk menjadi kecil agar tidak terlalu terpengaruh oleh hierarki *memory* dan lebih fokus pada pengujian *JVM* dan *CPU*.

Kernel yang dipilih adalah sebagai berikut:

- *Fast Fourier Transform (FFT)*

FFT melakukan transformasi satu dimensi dari *4K complex numbers*. Pengujian meliputi aritmatika rumit, *shuffling, non-constant memory references*, dan fungsi trigonometri. Bagian pertama menguji *bit-reversal portion* (*no flops*) dan bagian kedua melaksanakan N log (N) *computational steps*.

- *Gauss-Seidel relaxation*

Gauss-Seidel relaxation pada 100x100 *grid* menguji *access pattern* pada aplikasi lain yang terbatas

- *Monte Carlo integration*

Monte Carlo integration menghitung nilai Pi dengan cara menghitung hasil integral dari quarter circle y = sqrt(1 - x^2) pada [0,1]. Angka dipilih secara acak dengan satuan pangkat dua dan menghitung rasio dari angka tersebut dalam sebuah lingkaran. Algoritma ini menguji penghasil angka acak, pemanggilan fungsi secara sinkron, dan *function inlining*.

- *Sparse matrix multiply*

Sparse matrix multiply mengunakan matriks longgar tidak terstruktur yang disimpan pada format *compressed-low* dengan struktur kelonggaran yang telah ditentukan. *Kernel* ini menguji *indirection adressing* dan *non-regular memory references*. Matriks longgar 1000 kali 1000 dengan 5000 *nonzero* digunakan.

- *Dense LU matrix factorization*

Dense LU matrix factorization menghitung faktorisasi LU dari matriks 100 kali 100 menggunakan *partial pivoting*. Kernel ini menguji kernel aljabar linier dan operasi matriks padat[5].

2.3 *DriveSpeed2*

DriveSpeed2 bertujuan untuk mengukur kemampuan dari *SD Card* dan *internal drive*. *DriveSpeed* melakukan empat pengujian:
1. *Write* dan *read* tiga data 8 dan 16 MB. Hasil berupa MBytes per detik.
2. Menulis data 8 MB, membaca dan *cached* di RAM. Hasil berupa MBytes per detik.
3. Penulisan dan pembacaan acak data 1 KB dari 4 hingga 16 MB. Hasil berupa rata-rata dalam milidetik.
4. *Write* dan *read* 200 file 4 KB hingga 16 KB. Hasil dalam MBytes per detik, mili detik per file, dan waktu penghapusan.

DriveSpeed dibuat pertama kali pada awal mula berkembangnya *device Android* yang memiliki *internal drive* dan *SD card* eksternal. Dengan munculnya versi terbaru dari *Android, internal drive* bisa memiliki porsi kerja yang paling besar sebagai *drive* dan merangkap sebagai *cached drive*. *Directory path* yang digunakan juga bervariasi untuk setiap sistem[4].

2.4 *SunSpider*

SunSpider merupakan *JavaScript benchmark* yang menguji operasi *JavaScript* umum maupun operasi yang akan populer di masa yang akan datang seperti enkripsi dan manipulasi teks. Tes ini berfokus pada masalah-masalah di dunia nyata yang dihadapi oleh pengembang JavaScript saat ini dan masa yang akan datang. Pengujian meliputi *generate tagcloud* dari inputan *JSON, 3D raytracer,* kriptografi, dekompresi kode, dan lainnya. Pengujian yang menguji berbagai area dari bahasa dan berbagai tipe kode. Tidak hanya angka, *string*, atau *looping dasar*.

Pengujian ulang dilakukan untuk memastikan hasil sudah benar-benar mewakili dari *browser*. *Confidence interval* berada pada rentang 95%[6].

2.5 *OpenGL ES (OpenGL for Embedded System)*

OpenGL for Embedded System merupakan bagian dari *OpenGL API* (*Application Programming Interface*) untuk melakukan *render* grafik dua dimensi maupun tiga dimensi seperti pada *video games*. OpenGL ES merupakan *API* lintas bahasa dan *multi-platform* [3].

2.6 *JNI (Java Native Interface)*

JNI (*Java Native Interface*) merupakan sebuah mekanisme API yang memungkinkan program yang ditulis dalam bahasa *Java* melakukan pemanggilan *function* yang ditulis dalam bahasa C atau C++ dan sebaliknya program yang ditulis dalam bahasa C atau C++ dapat melakukan pemanggilan *method* dalam bahasa *Java*.

Tujuannya adalah memungkinkan perluasan fitur dalam sistem yang biasanya dijalankan lebih baik dalam bahasa C atau C++. Selain itu penggunaan JNI juga dapat dikaitkan dengan *performance* dari bahasa C atau C++ yang belum sepenuhnya dapat disamakan dengan *Java* maupun penggunaan ulang kode-kode dalam bahasa C atau C++ yang sudah teruji sebelumnya [7].

Dalam topik ini, penggunaan JNI ditujukan untuk menjalankan metode-metode *benchmarking* yang kebanyakan ditulis dalam bahasa C++ [1].

2.7 *RandMem*

RandMem test menguji kemampuan dari *RAM* (*Random Access Memory*) dari *device* dengan melakukan pengujian dalam ukuran data yang semakin besar untuk mencari kecepatan *transfer data* dalam satuan *MBytes Per Second* dari *cache* dan *memory*.

Pengujian dilakukan secara *serial* dan *random* dengan pengujian *read* dan *read write* menggunakan *integer* 32 *bit*. Tujuan dari pengujian ini adalah seberapa besar dampak dari akses data secara acak [4].

3. DESAIN SISTEM

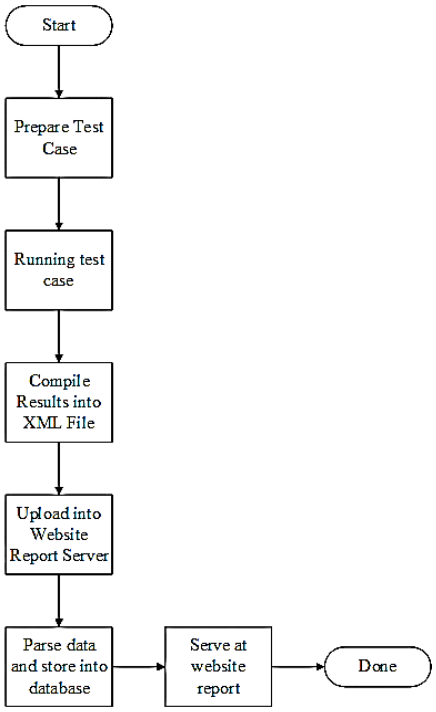
Dalam bab ini akan dijelaskan mengenasi desain sistem dari aplikasi SmaPon Benchmark. Desain sistam menjelaskan tentang gambaran sistem yang diterapkan dalam apliaksi.

3.1 Perencanaan *Software*

Berdasarkan analisis dari aplikasi *benchmark seperti AnTuTu, Quadrant Standard,* dan *Basemark OS II*, perlu dikembangkan sebuah aplikasi *benchmarking* yang mampu menjawab kebutuhan masyarakat Indonesia dalam memilih *device* yang sesuai. Aplikasi *benchmarking user friendly* dengan ukuran kecil dengan fitur utama yang tidak dikurangi, pemilihan metode yang mampu memberikan hasil yang mewakili *performance* dari *device*, penggunaan *web service* untuk kemudahan akses (*thin client*), penyajian data transparan, dan edukasi bagi pengguna. Untuk memastikan standar dan keabsahan *output*, tugas akhir ini menggunakan aplikasi *benchmark 0xbenchmark* sebagai acuan dasarnya..

3.2 *Flowchart* Sistem

Pada bagian ini akan dijelaskan tentang garis besar aplikasi *SmaPon Benchmark* yang dapat dilihat pada Gambar 1.



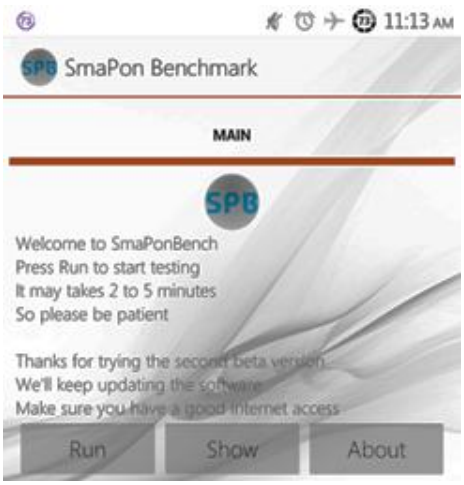
Gambar 1 Garis Besar Aplikasi

Pada Gambar 1 dapat dilihat garis besar aplikasi ketika program dibuka akan mempersiapkan *case* dan *scenario* pengujian. Ketika menekan tombol *Run* maka pengujian berjalan. Pengguna dapat melihat *detail* dari *device* melalui tombol *About* dan memeriksa hasil pengujian melalui tombol Show.

4. PENGUJIAN SISTEM

Pengujian terhadap aplikasi dan situs yang telah selesai dibuat dilakukan dengan cara melakukan proses secara keseluruhan. Hal ini bertujuan untuk mengetahui apakah aplikasi dan *web service* yang telah dibuat dapat berjalan dengan baik atau tidak. Berikut ini urutan tahapan dalam melakukan pengujian program yang telah dibuat. Seperti *Website* pada Gambar 2 yang menunjukkan 50 hasil terbaik pengujian aplikasi *benchmark* disertai dengan *chart* sebagai visualisasi data yang menampilkan 25 hasil terbaik dari *device* yang berbeda-beda dan layar utama aplikasi pada Gambar 3 yang menunjukkan *menu* utama dari aplikasi untuk memilih *activity* yang akan dijalankan.

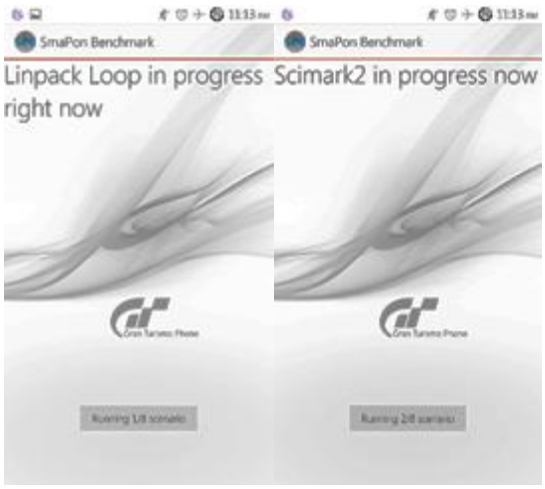
Gambar 2 Hasil Aplikasi *Website*



Gambar 3 Hasil Aplikasi Android

4.1 Pengujian *Test CPU*

Pengujian pertama adalah pengujian CPU dengan metode *Linpack*. Linpack dijalankan sebanyak tiga kali. Setelah itu pengujian akan menjalankan *activity* untuk metode pengujian *SciMark2*. Metode *SciMark2* menjalankan 6 *computational kernel* yang berbeda-beda.

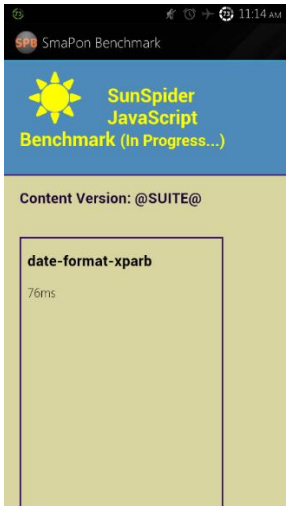


Gambar 4 *Activity Linpack* dan *SciMark2*

Pada Gambar 4 pengujian *LINPACK* (kiri) dan *SciMark2* (kanan) dapat berjalan dengan baik

4.2 Pengujian *Browser JavaScript Engine*

Setelah pengujian CPU, aplikasi akan secara langsung menjalankan *activity* untuk pengujian *JavaScript* yakni metode *SunSpider* berupa *WebView* yang menggunakan peramban web *built-in* dari *device*.

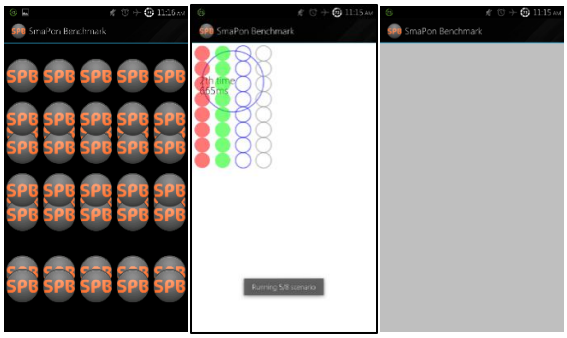


Gambar 5 *Activity SunSpider JavaScript Benchmark*

Pada Gambar 5, dapat dilihat aplikasi sedang melakukan pengujian *JavaScript* dengan *SunSpider*

4.3 Pengujian *2D Graphic*

Untuk pengujian grafis dua dimensi (2D) metode pengujian pertama adalah *Draw Canvas*. Berapa lama waktu yang diperlukan untuk dapat melakukan *redraw canvas* sebanyak 300 kali. Kedua *DrawCircle*, dan *DrawImage* seperti pada Gambar 6. *DrawCanvas* menghitung jumlah *frame per second* yang dapat dicapai saat melakukan *redraw canvas*. *DrawImage* melakukan perhitungan *frame per second* saat *device* melakukan *rendering image* dua dimensi saat dilakukan animasi transisi. *DrawCircle* menghitung *frame per second* saat *device* melakukan *rendering set* lingkaran yang telah ditentukan untuk menjaga agar tiap *device* diuji secara *fair*.



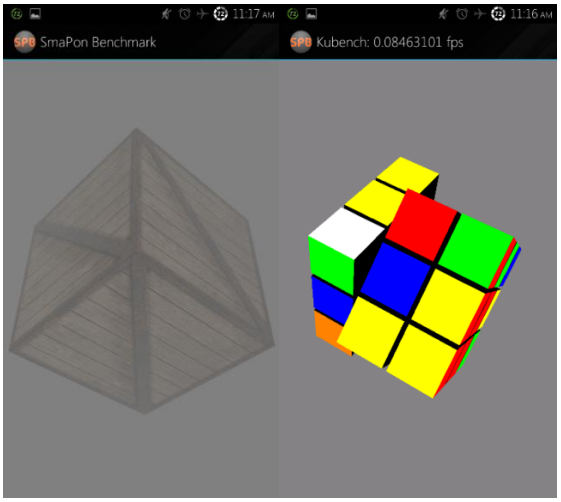
Gambar 6 *Activity* Pengujian Grafis 2D

Pada Gambar 6, *DrawImage* (kiri), *DrawCircle* (tengah), dan *DrawCanvas* (kanan) berjalan dengan baik.

4.4 Pengujian *3D Graphic*

Untuk pengujian grafis tiga dimensi (3D), metode pertama yang digunakan adalah *DrawCube* dari *Kubench* yang merupakan *demo OpenGL ES* untuk *Android*. Aplikasi akan melakukan *render* sebuah kubus *Rubik* yang bergerak. Akan dihitung *framerate* dengan satuan *frame per second* (FPS).

Activity pengujian grafis 3D yang terakhir adalah *OpenGL Fog* yang melakukan *rendering* sebuah *crate* dalam lingkungan yang berkabut. Metode ini diambil dari *Tutorial NeHe*. Output yang diperoleh berupa *framerate* dalam satuan FPS.



Gambar 7 *Activity* pengujian grafis 3D

Gambar 7 menunjukkan *activity* pengujian *OpenGL Fog* (kiri) yang menghiting *framerate device* saat melakukan *rendering box* ber-*texture* pada *environment* yang dipenuhi oleh kabut. *Activity OpenGL Cube* menghitung *framerate device* saat melakukan *rendering* kubus *Rubik* yang sedang menyelesaikan *puzzle* tersebut.

4.5 Laporan Hasil Pengujian

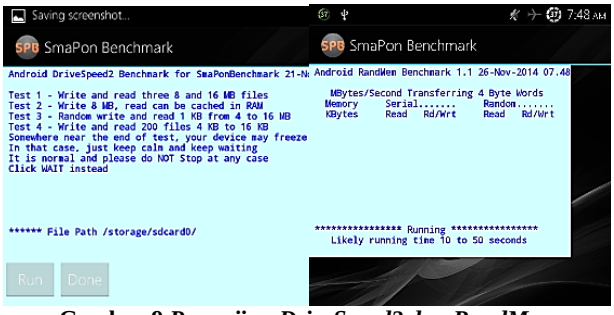
Setelah pengujian selesai, aplikasi masuk pada halaman laporan hasil pengujian. Pada *activity* tersebut, hasil mentah dari pengujian ditampilkan. Akan tetapi pengguna belum dapat

menggungah hasil pengujian sebelum pengujian *Storage*. Pengguna dapat hanya melakukan analisa dari hasil asli dari pengujian dan mengakhiri proses pengujian dengan menekan tombol "*Done*". Seperti pada Gambar 8.

Gambar 8 *Activity Report*

4.6 *Native Test*

Selanjutnya pengujian berlanjut dengan metode *DriveSpeed2* dan *RandMem* yang bertujuan untuk melakukan *benchmarking* untuk *storage* dan *RAM* dari *device*

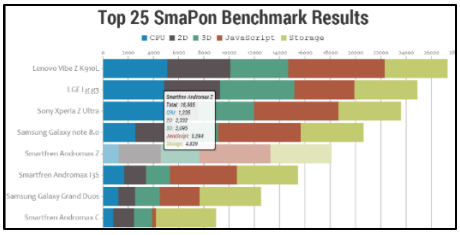


Gambar 9 *Pengujian DriveSpeed2 dan RandMem*

Dari hasil pengujian dapat dilihat pada Gambar 9 bahwa pengujian berjalan dengan baik.

4.7 Pengujian *Upload* Hasil *Benchmark*

SmaPonBench menggunakan file *XML* (*Extensible Media Language*) sebagai media untuk pengiriman data pada *server*. *Server* menyimpan *file XML* yang telah diunggah dari *device*. Jika ada hasil pengujian dari *device* yang sama maka akan dibandingkan hasilnya dan akan menentukan apakah hasil pengujian akan ditampilkan pada *table* utama pada *web service*. Seluruh data hasil pengujian akan tetap disimpan pada *table*.



Gambar 10 *Chart* Hasil Pengujian Pada Web Service

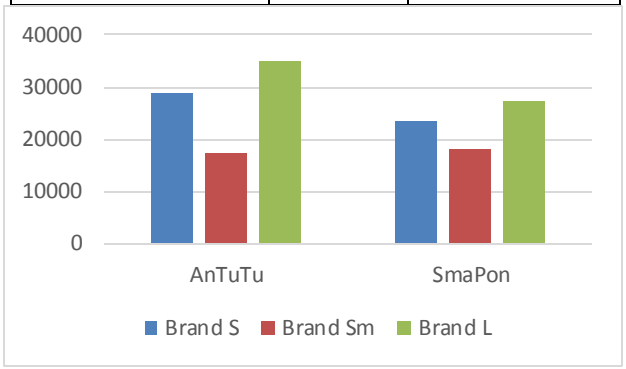
Dari hasil pengujian dapat dilihat pada Gambar 10 bahwa hasil pengujian telah berhasil ditampilkan pada *web service* berupa *chart* hasil pengujian *device*.

4.8 Perbandingan Dengan Aplikasi Serupa

Hasil pengujian dari *AnTuTu* dan *SmaPon* dapat dibandingkan menunjukkan kemiripan dalam hasil dalam perbandingan hasil antar *device* pada aplikasi *AnTuTu* 4 dan *SmaPon Benchmark*.

Tabel 1 Komparasi Hasil SmaPon dan AnTuTu 4

Device	AnTuTu 4	SmaPonBench
Brand S	28965	23596
Brand Sm	17484	18085
Brand L	35106	27229



Gambar 11 Chart Komparasi SmaPon dan AnTuTu

Hasil perbandingan antara kedua aplikasi menunjukkan *trend* yang sama. Seperti pada Tabel 1 dan Gambar 11. Untuk *device Brand L* dengan spesifikasi *hardware* terbaik pada pengujian ini, *device* ini menghasilkan *output score* terbaik. Kemudian untuk *device Brand Sm* yang merupakan *device mid-end* dengan kapabilitas *hardware* yang kurang disbanding kedua *device* lain menunjukkan *output score* paling rendah pada pengujian ini.

4.9 Pengujian Dalam Berbagai Kondisi

Pengujian dilakukan pada beberapa kondisi untuk menguji konsistensi dan relevansi *output* hasil pengujian. Pengujian dilakukan berdasarkan waktu *boot* dan *free available RAM*. Berikut merupakan hasil pengujian berdasarkan waktu *boot*. Pengujian dilakukan pada *device Brand Sm* dengan *maximum available RAM*.

Tabel 2 Hasil Pengujian Pada Waktu *Boot* Berbeda

Waktu Terakhir *Boot*	Hasil Total Pengujian
Fresh boot	18085
1 Jam	17883
3 Jam	17892
5 Jam	17793
10 Jam	17829

Berdasarkan hasil pengujian, tampak *trend* menurun sejak *device* pertama kali menjalankan pengujian saat *bootup*. Hal ini dapat dihubungkan dengan beberapa aplikasi pada *device* melakukan *caching* secara berkala sehingga tidak banyak porsi tersedia untuk aplikasi *benchmarking*.

Seperti yang terlihat pada Tabel 2, semakin lama *device* menyala maka hasil pengujian tampak semakin menurun hasilnya.

Untuk melakukan simulasi penggunaan *RAM*, *device* dibebani dengan menjalankan aplikasi default dari *device* seperti aplikasi *SMS*, *Phone Dialer*, *Calendar*, *Contacts*, *Note*, *Browser*, *Clock*, *Settings*, dan *Camera*. Untuk menghindari pencemaran akibat lama waktu *boot*, tiap pengujian dilakukan dengan melakukan *fresh boot.*

Tabel 3 Hasil Pengujian Pada *Free Available RAM Berbeda*

Free Available RAM	Hasil Total Pengujian
75%	18085
60%	17983
45%	17862
30%	17154
20%	16929

Berdasarkan hasil pengujian pada Tabel 3, jumlah *RAM* yang dapat digunakan oleh *device* sangat berpengaruh terhadap hasil pengujian. Hal ini dapat disebabkan oleh metode pengujian *RandMem* berdampak langsung pada hasil pengujian. Selain itu metode pengujian SciMark 2.0 juga terpengaruh oleh *RAM*.

5. KESIMPULAN

Berdasarkan hasil pengujian dapat disimpulkan beberapa hal sebagai berikut:

- Berdasar hasil pengujian, aplikasi dapat berjalan dengan baik dan dapat menjalankan semua fungsi - fungsinya.
- Hasil dari proses *benchmark* relevan dengan kapabilitas dari *hardware*. Hasil *benchmark* juga tidak melenceng jauh dari aplikasi serupa.
- Kondisi *device* sangat mempengaruhi hasil pengujian, jumlah RAM, jumlah sisa *cache*, kondisi temperatur CPU, dan ruang tersisa pada *SD Card* karena metode pengujian menggunakan seluruh *resource* dari *device*. Beberapa metode malah menguji bagian spesifik seperti *RAM* dan *Storage*.
- Untuk beberapa *device* dengan sistem operasi *Android* yang lebih baru (4.4 KitKat) terdapat batasan perihal penggunaan *storage* yang mengakibatkan pengujian *storage* tidak dapat dilakukan.
- *Device* dengan *hardware capability* tinggi dapat menghasilkan *performance* yang lebih baik.
- Dengan penggunaan *resource* yang kecil yang berujung pada kecilnya ukuran aplikasi secara keseluruhan membuat proses *download* dari aplikasi jadi lebih cepat dan mudah. Menyisakan *resource* lebih banyak untuk proses pengujian, sehingga hasil pengujian dapat lebih maksimal

6. DAFTAR PUSTAKA

[1] Chuan, H. C. 2014. *Java Programming Tutorial Java Native Interface (JNI)*. Retrieved from http://www3.ntu.edu.sg /home/ehchua/programming/java/JavaNativeInterface.html

[2] Donggara, J., Luszcek, P., & Petiet, A. 2002, The LINPACK Benchmark: Past, Present, and Future, *Exper* (15), 803-820. doi: 10.102/cpe.728.

[3] Khronos. 2014. *The Standard for Embedded Accelerated 3D Graphics*. Retrieved from https://www.Khronos.org/ opengles/

[4] Longbottom, R. 2014. *Android Benchmarks*. Retrieved from http://www.roylongbottom.org.uk/android%20benchmarks.htm

[5] Pozo, R. 2004. *SciMark 2.0 Java Benchmarking*. Retrieved from http://math.nist.gov/scimark2/about.html

[6] Webkit. 2014. *SunSpider 1.0.2 JavaScript Benchmark*. Retrieved from https://www.webkit.org/perf/sunspider/ sunspider.html

[7] Lee Y. H., Chandrian. P., & Li. B. 2011. *Efficient Java Native Interface for Android Based Mobile Devices*. *Proceedings of the 2011IEEE 10th International Conference on Trust, Security, and Privacy in Computing and Communications*, TRUSTCOM 2011, 1202-1209, Washington, DC, USA, 2011. IEEE Computer Society.

[8] Yoon, H. J. 2012. *A Study on the Performance of Android Platform*, *International Journal on Computer Science and Engineering*, 4(4). 532-537. Retrieved from http://www.enggjournals.com/ijcse/doc/IJCSE-04-04-128.pdf